A Software Model for Visualization of Time Dependent 3-D Computational Fluid Dynamics Results

Al Globus, Computer Sciences Corporation

Report RNR-92-031

9 November 1992

NASA
National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

ARC 275a (Feb 81)

A Software Model for Visualization of Time Dependent 3-D Computational Fluid Dynamics Results

Al Globus, Computer Sciences Corporation

Report RNR-92-031

9 November 1992

This work was supported by NASA Contract NAS 2-12961 to Computer Sciences Corporation for the Numerical Aerodynamic Simulation Systems Division at NASA Ames Research Center.

Copies of this report are available from:

NAS Applied Research Office
Mail Stop T045-1
NASA Ames Research Center
Moffett Field, CA 94035
(415) 604-4332

A Software Model for Visualization of Time Dependent 3-D Computational Fluid Dynamics Results

Al Globus, Computer Sciences Corporation[1]

Abstract

We believe that the current generation of scientific visualization systems are inadequate for flexible, interactive, animated visualization of the largest time dependent numerical simulations. The current approaches are inherently too inflexible and/or too slow. To address this deficiency, we propose a new software model for time dependent 3-D visualization. The model introduces concepts to improve visualization of very large (many gigabyte) time-dependent numerical solutions. To provide direct user control over crucial time vs. space vs. flexibility trade-offs, the model includes a persistent object database of all scientific and visualization data including time and space metrics. The model provides a common framework for users and developers to understand visualization system requirements and capabilities. There are currently a number of efforts to incorporate the model into existing and new visualization systems.

I. Introduction

The CFD (computational fluid dynamics) community has used visualization systems extensively for many years [Buning85]. CFD results may be divided into time-independent and time-dependent data sets. Among recent 3-D CFD results at NAS[2], time independent data sets range in size from quite small to about 80 megabytes while the largest time dependent data sets range from 5,600 to 162,000 megabytes (see table 1[3]).

Table 1: A Few Time Dependent CFD Simulations at NAS

Data Set	Grid Nodes	Time Steps	Time Step Size (megabytes)	Total Size (megabytes)
Smith91	2,800,000	100	56	5,600
GB92	1,400,000	400	28	11,200
Atwood92	1,600,000	500	32	16,000
Chawla92	900,000	9,000	18	162,000

1. This work is supported through NASA contract NAS2-12961.
2. The NAS (Numerical Aerodynamic Simulation) Systems Division at NASA Ames Research Center is a leading supercomputer facility dedicated to the study of computational aerosciences, particularly CFD [Bailey86].
3. Section V has a more complete version of this table.

Many visualization systems are successfully used to examine time independent CFD results [e.g., Buning85, Upson89, Bancroft90, Legensky90, Wavefront91, SGI92]. However, visualization of 3-D time dependent results is currently very difficult due to the sheer quantity of data. When faced with data quantity problems, it is tempting to wait for hardware improvements. However, hardware improvements inevitably imply even larger CFD results. Thus, we believe visualization system software requires major improvements to convenient and flexibly visualize the largest time dependent CFD results.

We propose a software model to guide visualization system evolution. Since data set size is the core of the problem, the model focuses almost exclusively on the data, intelligent data compression, database issues, and time vs. space vs. flexibility trade-offs. Most visualization systems implement parts of this model, but we believe that all components are crucial.

Previous Work

Smith89 proposed unsteady visualization by taking subsets of the data and visualizing the subsets rather than the entire data set. Smith, et. al., write "... in many cases the storage of the solution set versus time at a few well-chosen cross-sections (e.g., two-coordinate slices through the three-coordinate space) can capture the critical flow features." This is the core of our extract concept (see below).

Levit and Krystynak are currently completing a 2-D time dependent flow visualization package called CPT [Levit92]. While this is an excellent system, the approach will not generalize to 3-D time dependent flows where data management problems are much more severe.

Haimes91 has developed the Visual3 software package to visualize unsteady 3-D flows, particularly for unstructured grids. Time controls are, however, primitive. Dickenson91 presents excellent time control facilities. Bryson91 uses virtual reality techniques to examine time-dependent vector fields. Yamasaki92b distributes computation between supercomputer and workstation to visualize time dependent flow.

To our knowledge, no existing general-purpose visualization system has demonstrated results on the largest unsteady data sets calculated from solutions at speeds sufficient to give the illusion of motion[1], although Bryson91 and Yamasaki92b have succeeded with a limited set of visualization techniques on a somewhat smaller data set [Levit91].

Overview

Section II introduces the model. Sections III and IV take an in depth look at two major aspects of the model neglected by current visualization systems. Section V looks at time vs. space vs. flexibility trade-off issues. Section VI is a general discussion of the model's uses and implications. Section VII is our future plans. Section VIII is a brief summary. Appendix A is a glossary. There are two appendixes examining current visualization practice in the model's terms. Appendix B discusses visualization techniques while appendix C discusses existing visualization systems. Appendix D contains data relevant to the space vs. time trade-offs discussed in section V.

1. Usually at least ten frames per second. Note that video rates, thirty frames a second, are not generally needed for scientific visualization.

II. The Model

Visualization systems[1] input raw data and output images.

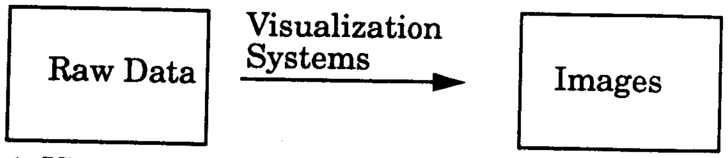


Figure 1: Visualization Systems Convert Raw Data into Images

For CFD and many other numerical simulations, raw data are solutions of partial differential equations represented by scalar and vector fields sampled on volumetric grids. For time-dependent solutions, time is sampled discretely; i.e., the scalar and vector fields, and in some cases the grid, change with each time step. As we are concerned with visualizing the largest solutions, for the purpose of this paper solutions are 3-D and time-dependent unless otherwise noted.

Images are very inflexible - the viewpoint cannot be changed. Current visualization systems generally extract data structures describing 3-D graphic objects from static solutions. Modern workstation hardware can convert these graphic objects to images fast enough[2] for interactive viewpoint control. A *graphic object* is a set of vertices combined with rendering information such as vertex color and normal vector. The terms *display list*, *geometry*, and *geom* are approximate synonyms.

We see that most visualization systems support only two primary types of data: solutions and graphic objects. A solution is transferred from disk to RAM then graphic objects are extracted by visualization algorithms from the solution and rendered. This model has proven adequate for visualizing large static 3-D CFD solutions and time-dependent 2-D solutions.

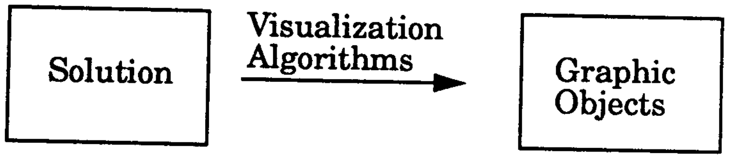


Figure 2: Traditional Visualization Model

To visualize time-dependent 3-D solutions, smooth animation[3] of successive solution time steps is required and interactive response is highly desirable. For the largest solutions, this is difficult or impossible using the traditional model because the largest solutions will not fit in RAM, and sometimes not even on disk. Furthermore, although graphic objects allow interactive viewpoint control, they are otherwise relatively inflexible. For example, the number of contour lines representing a scalar surface cannot be changed without recalculation from the solution.

The key to visualizing enormous time-dependent solutions is to add an intermediate data type between solution and graphic objects called *extracts*. An extract is a subset or resampling of the solution's scalar and vector fields. Converting a solution to extracts achieves substantial, albeit lossy[4], data compression by reducing data dimensionality or reducing the size of space or time

1. By visualization system we mean software designed to visualize scientific data in a general way using many different techniques within a coherent framework.
2. At least a few times a second.
3. Usually requiring eight or more frames per second.

dimensions. If storage is limited, solution time steps may be archived or deleted once the proper extracts have been extracted. Choosing the right extracts is critical to successful visualization.

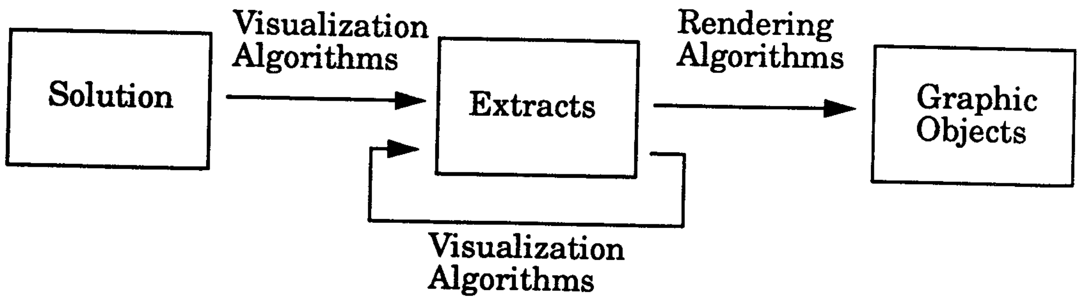


Figure 3: Extracts Provide Flexibility and Lossy Data Compression

Most visualization algorithms are easily modified to produce extracts rather than graphic objects by deferring rendering decisions. For example, a grid plane extract is a subset of a solution grid's nodes. Resampling is illustrated by a marching cubes [Lorenson87] isosurface extract. The isosurfaces's vertices are the resampling points.

Extracts may be represented by grids with field data at each grid node. This contrasts with graphic objects, which are vertices with rendering information. Obviously, graphic objects are closely related to extracts. Consider an isosurface graphic object. Isosurfaces are generally represented by lists of polygons, usually triangles, with normal vectors at each vertex. An unstructured surface grid consists of a list of polygons, usually triangles. Any set of scalars, vectors, or tensors could be stored at each grid node. Thus, extracts differ from graphic objects by storing data rather than rendering information at grid nodes (vertices). This substantially increases flexibility without requiring additional storage.

Most extracts may be rendered in many ways. Consider a streamline[1] with a scalar field at each vertex. This extract can be rendered as a color mapped curve, a stream tube [Schroeder91] with diameter as a function of scalar value, a ribbon with ribbon orientation controlled by the scalar field, etc. Note that extract rendering decisions can be made or changed even after the solution has been deleted.

For additional flexibility over graphic objects, one may calculate one extract from another. Consider a velocity field on a cut-plane. Streamlines constrained to the plane can be calculated from a cut-plane extract without referencing the solution.

Time step animation is crucial for visualizing time-dependent solutions. A minimal animation data type is a sequence of images, e.g., a video tape. We call this a *movie*. Like images, movies have no viewpoint control. To provide viewpoint control we also provide *scenes*. A scene is a set of graphic objects evolving through time and a *navigation*. A navigation is the path a virtual camera might take through time and space to create a movie, i.e., an ordered set of viewpoints and

4. Lossy data compression does not preserve all of the information; i.e., one cannot reconstruction the original data from the compressed form.

1. A streamline is a particle trace in a steady velocity field.

viewing directions. A movie is created by 'turning on' a virtual camera and running a scene.

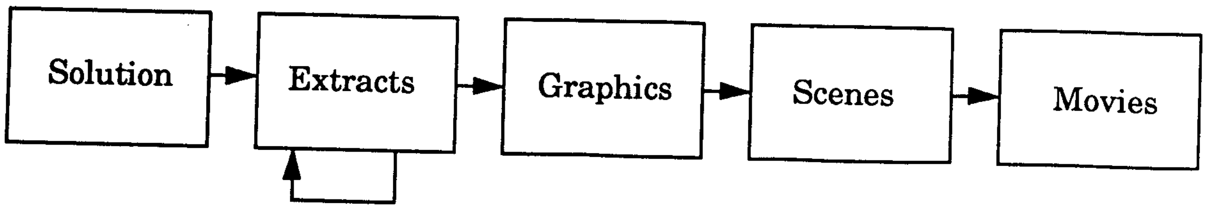


Figure 4: Animation Facilities are Required

Each solution, extract, graphic object, scene, navigation, and movie is an object. A database containing all objects is needed to provide:

- Unified access to all objects. This is a major convenience and is necessary to implement time vs. space trade-offs by deleting objects.
- Performance metrics, i.e., object generation CPU time and storage requirements. These are essential to making informed trade-offs.
- Save and restore. Essential to avoid losing work when an application must exit.
- Object history. A description of how each object was created. This is useful to conveniently implement repetitive operations[1] and for making time vs. space trade-offs when saving state; e.g., choosing whether to save an isosurface to disk or recalculate it when restoring state.

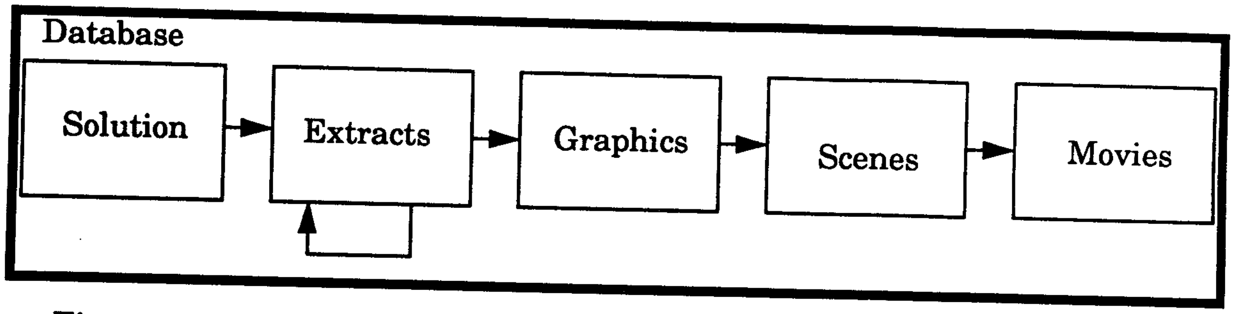


Figure 5: A Database Provides a Unified Interface and Time/Space Metrics

To complete the model, we add software tools to manipulate each of the data structures and the database as a whole. We deliberately leave tool definition for last to minimize the importance of

1. For example, calculating an isosurface for many time steps.

the tools and focus attention on the data.

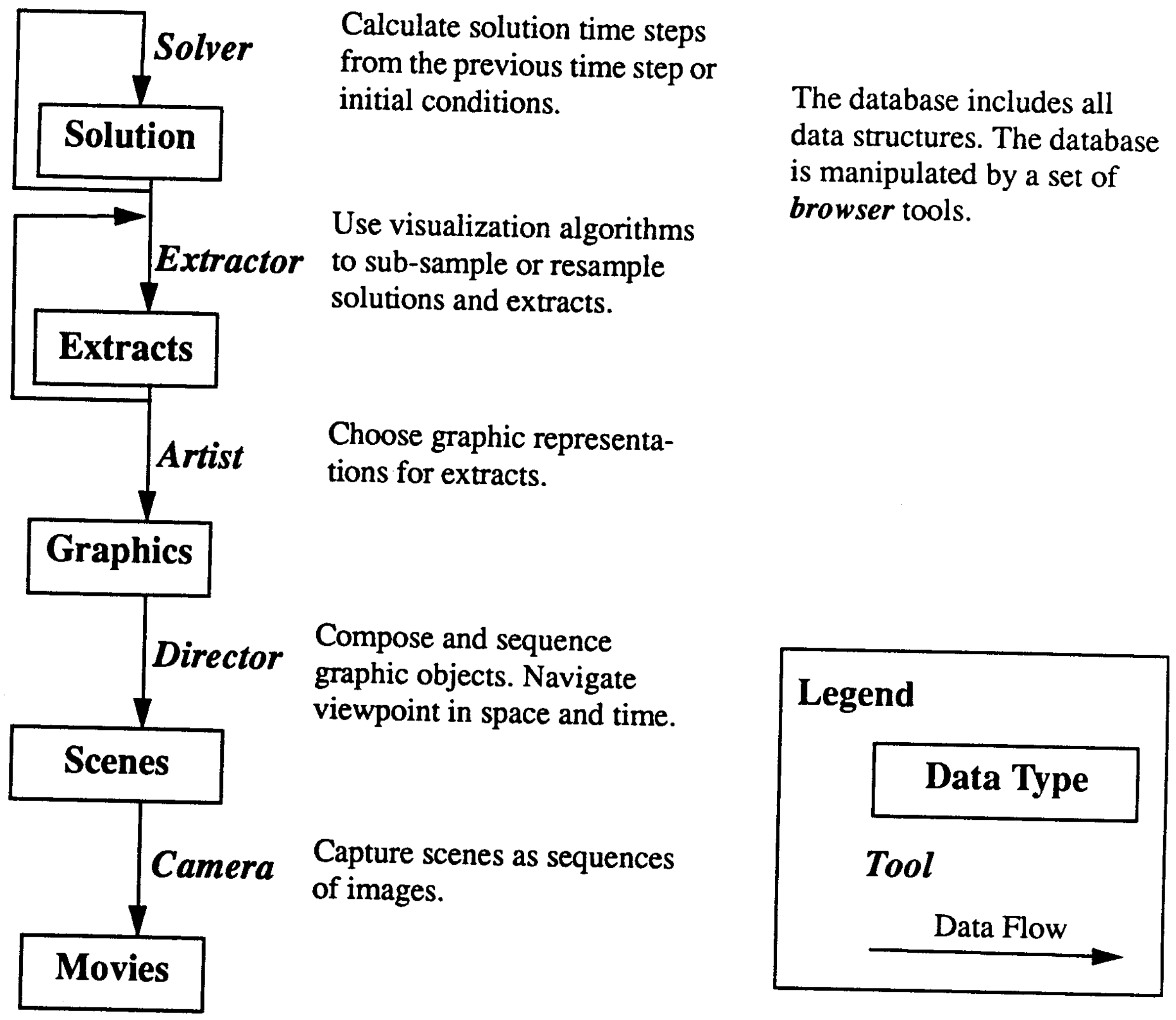
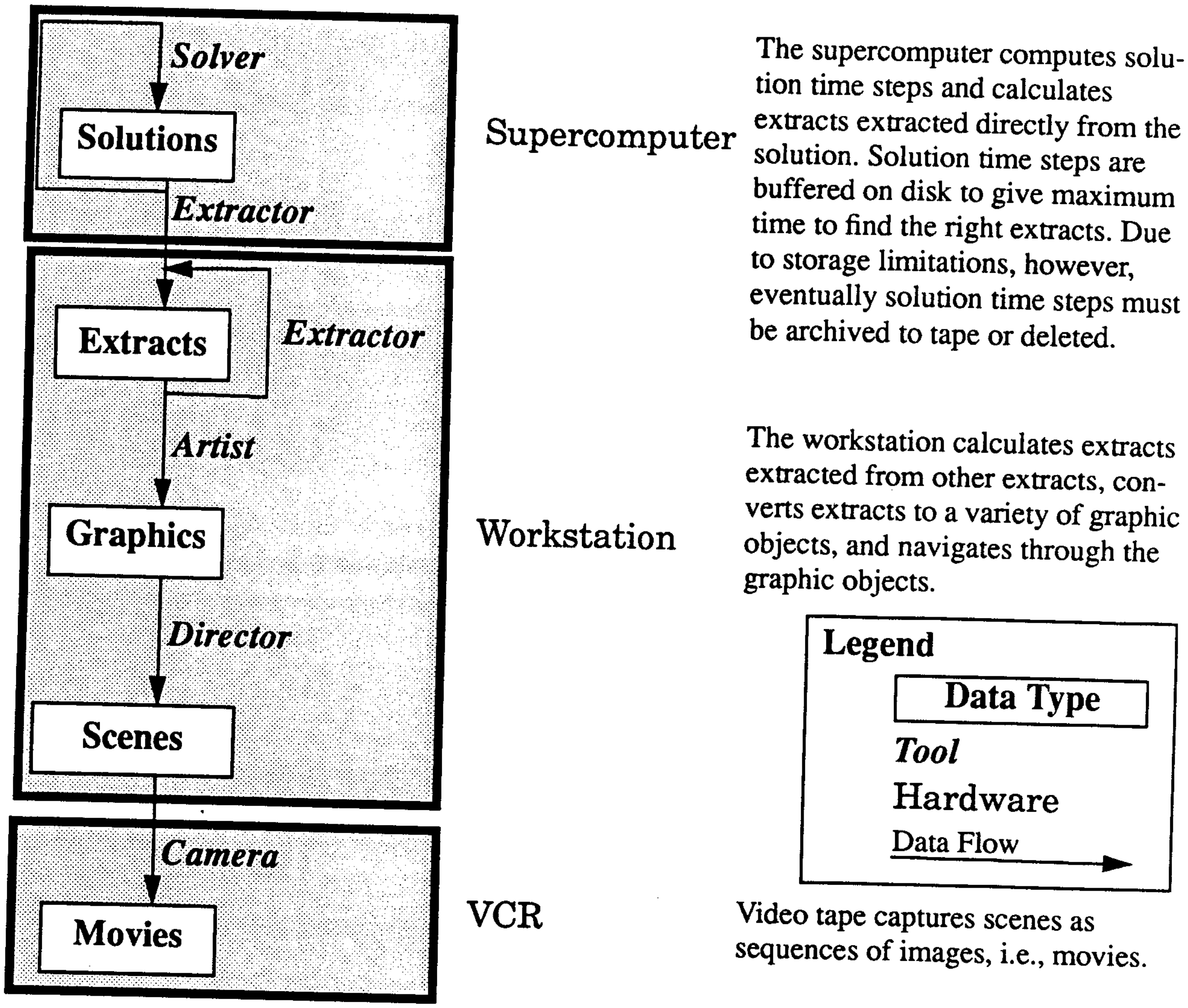

Solver
Calculate solution time steps from the previous time step or initial conditions.
Solution
Extractor
Use visualization algorithms to sub-sample or resample solutions and extracts.
Extracts
Artist
Choose graphic representations for extracts.
Graphics
Director
Compose and sequence graphic objects. Navigate viewpoint in space and time.
Scenes
Camera
Capture scenes as sequences of images.
Movies
The database includes all data structures. The database is manipulated by a set of *browser* tools.
Legend
Data Type
Tool
Data Flow

Figure 6: Tools Used to Create Data Structures

The terms artist, director, and camera are borrowed from the film industry. An artist chooses a visual representation for an idea (extract). A director chooses props and actors (graphic objects) and camera angles (navigation). Cameras record movies.

Although the model is very flexible and can be used in many ways, it was developed to address the situation where the solution is calculated on a supercomputer, visualized on a workstation, and there is not enough disk space to store all solution time steps. An approach to this problem using the model is illustrated in figure 7. Note that the database functionality is needed to manage the storage and CPU resources on each machine, deleting and transferring objects as necessary.



The supercomputer computes solution time steps and calculates extracts extracted directly from the solution. Solution time steps are buffered on disk to give maximum time to find the right extracts. Due to storage limitations, however, eventually solution time steps must be archived to tape or deleted.

The workstation calculates extracts extracted from other extracts, converts extracts to a variety of graphic objects, and navigates through the graphic objects.

Video tape captures scenes as sequences of images, i.e., movies.

Figure 7: One Use of the Model

With this approach, it is possible to visualize solutions that are much larger than available storage if the desired extracts are known *a priori*. Since solvers generate time steps sequentially and slowly (see table 4), extracts may be extracted from solution time steps as they are produced. Once extraction is complete, the solution time step is discarded. Extracts are generally much smaller than solutions, so extracts can be chosen to fit in available storage[1]. If the required extracts are not known *a priori*, solution time steps may be buffered and examined to choose extracts until storage runs out -- at which time old solution time steps must be discarded (or archived) to make room for the new. When solution time steps must be discarded, choosing the proper extracts is crucial to successful simulation, just as choosing the proper type and location of sensors is crucial to successful windtunnel experimentation. Choosing the proper extracts is much easier if a numerical experiment is designed to test a specific hypothesis, rather than to 'simulate system XYZ.'

1. The space vs. time vs. flexibility trades involved in this decision are discussed in section V.

We believe that extracts and the database are necessary to address the data management problems that must be solved to visualize the largest time dependent CFD data sets. Unfortunately, these elements are not emphasized in existing visualization systems. We discuss these elements of the model in the next two sections.

III. Extracts

Extracts are fields on grids. Extract grids may be 3, 2, 1, or 0-D objects embedded in $\mathcal{R}^3$ evolving from time step to time step. Two-D grids represent surfaces, 1-D grids represent curves, and 0-D grids represent points (for example, a number of disconnected particles released in a flow). While there are a plethora of common visualization techniques that generate graphic objects, removing the rendering issues irrelevant to extracts substantially reduces the number of techniques. Some of them are[1]:

- A subset of a solution's nodes in time and space; e.g., a single plane of a 3-D curvilinear grid, the exterior surface of an unstructured grid, or all nodes with a scalar field value above a threshold.
- Marching cubes [Lorenson87] generated surfaces, usually isosurfaces or cutting planes [Kerlick89]. These grids are unstructured surface (2-D) grids.
- Integration. Streamlines, particle traces, particle paths, streaklines, and tangent curves, are generated by integrating particles through a vector field. If particles are integrated within a single time step, 1-D grids are produced. If particles are integrated though time, 0-D grids are produced. Stream surfaces are produced by integrating a curve through a vector field at each time step.
- Ray casting. Although most ray casting algorithms produces images as output, Haimes91 uses a x-ray model to produce scalar fields on grids embedded in $\mathcal{R}^3$, i.e., extracts.
- Vector field stationary points[2], i.e., those points where a vector field vanishes. These points are crucial to vector field topology visualizations [Helman90, Globus91a]

Any number or type of fields may be calculated and stored on an extract's grid, but there is one 'field' that has very special properties. This 'field' is known as computational space. Computational space values are pointers into the solution's data structures along with interpolation coefficients.

To understand computational space, note that volumetric grids divide $\mathcal{R}^3$ into cells. A cell has an interpolation scheme, usually linear, that defines the field at all points within the cell. A computational space value is a cell identifier along with interpolation coefficients. For curvilinear grids, a computational space value can be three floating point numbers. The integer part specifies a cell (usually the lower left vertex of a hexahedron) and the fractional part specifies the interpolation coefficients. If an extract saves the computational space field on its grid, new fields may be calculated from the solution by simple interpolation. This improves performance when solutions and extracts have static grids and the extract's field must be calculated for many time steps. Storing the computational space field on an extract also allows deferred decisions on the fields to calculate. Finally, note that the computational space field can even be used to calculate the $\mathcal{R}^3$ values of the grid's nodes! [Haimes92] Thus, an extract need only produce the computational space field.

1. See appendix C for further explanation of each technique.
2. A.K.A. critical points.

All other fields, including the location of grid nodes in physical space, may be calculated later as long as the solution is preserved.

IV. The Object Database

Visualization using the model creates many disparate data objects. Thinking of all of these objects as parts of a coherent object database provides unity and completeness to a visualization system. In other words, all objects may be accessed through at least one common interface. Furthermore, the database can be made persistent, i.e., can be saved on disk without loss of state, so that visualization work sessions can be restarted, ported to new machines, and shared with others. Once a database exists, it is possible to collect information that supports time vs. space vs. flexibility trade-off decisions. Specifically, object time and space metrics and histories should be maintained.

Resource metrics, i.e., time and space to create or maintain objects, are needed to make intelligent trade-offs allocating scarce computational resources to visualization tasks. E.g., the storage required for an isosurface not currently being rendered vs. the time to re-calculate the isosurface from the solution; or the flexibility lost when a solution is discarded after extracts are calculated vs. the storage required by the solution.

In many cases, an extract must be calculated from many solution time steps. Objects that know their history (he data and procedures used to calculate an object) can be recalculated when the data they are derived from change, e.g., when another time step becomes available from the solver or disk. In addition, facilities to automatically recover objects can save significant rework. When an object is deleted, instructions for regenerating the object can be saved along with estimates of regeneration time and the storage required. To implement recovery, objects must know their history.

Considerable effort may go into creating a database. It should not be lost simply because the application exits. A database can be stored on a persistent media, such as disk, when a user quits the visualization system. Since objects are aware of their history and resource requirements, it is possible to make trade-offs between storing all objects as data or recalculating. For example, if disk space is at a premium, most objects can be stored as instructions for regeneration or if disk space is plentiful and users are pressed for time, most objects can be stored as data. In many cases one needs to save a portion of a database and import it elsewhere. Thus, the ability to select a portion of the database for export or add a portion of a database to another is important.

V. Time vs. Space vs. Flexibility Trade-Offs

On any computer system there is some hard limit to the amount of data that can be stored. There are also limits to the amount of time users are willing to spend on a given task, although these limits are less well defined. Of course, it is frequently possible to trade space for time and vice versa. When visualizing the largest numerical solutions, time and space limits quickly become constraints. Successful visualization requires an understanding of the issues driving space vs. time trade-offs. In particular, when space limitations are reached something must be deleted. This involves loss of flexibility, or, if the deleted object can be recalculated, a time trade-off. Figure 8 illustrates the general space, time, and flexibility characteristics of the model's data structures.

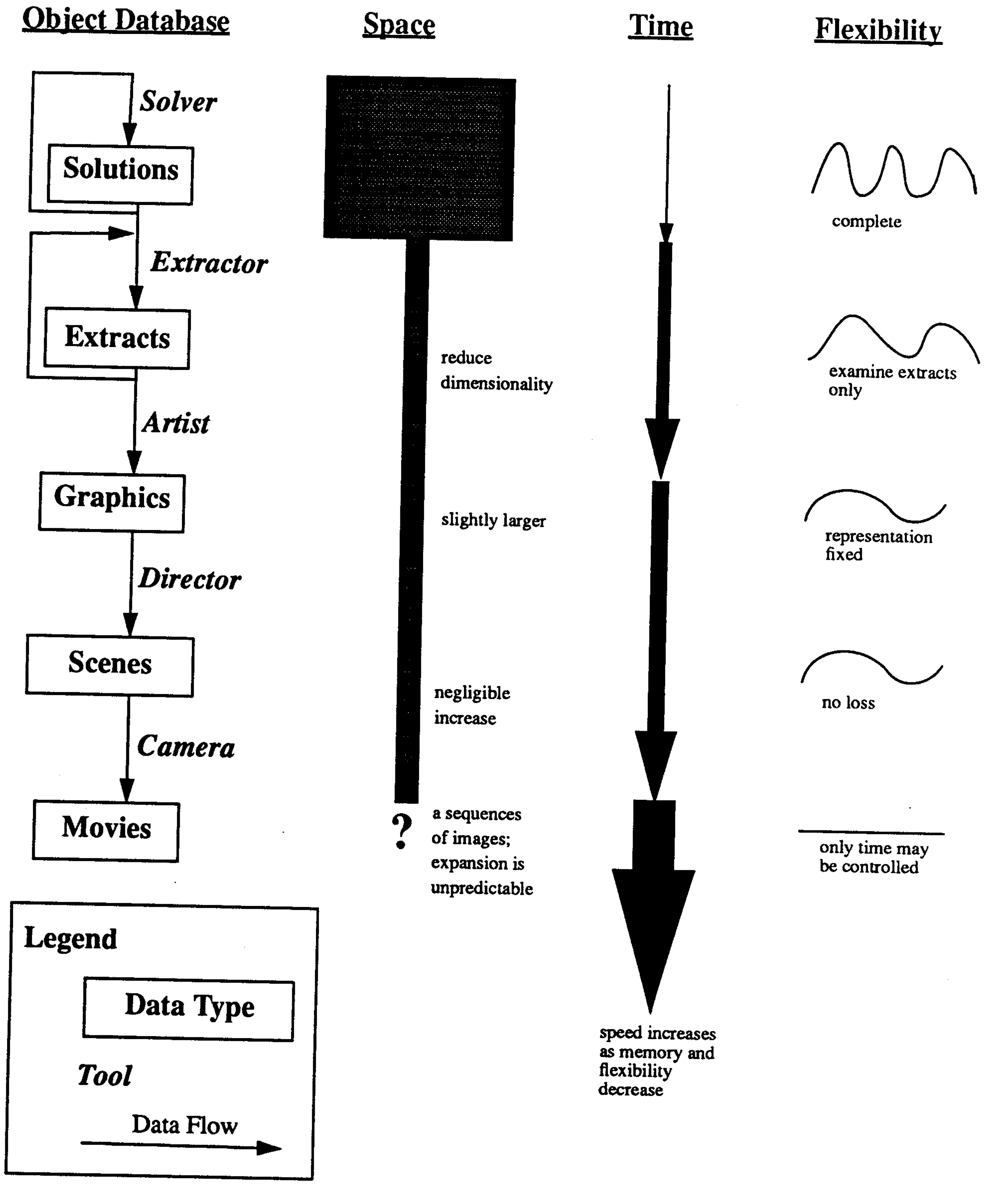


Figure 8: Time vs. Space vs. Flexibility

The dominant fact is that solutions are large and take a long time to calculate. Solutions are so large that even reading stored solutions from disk can take considerable time (see table 4 in

appendix D). Solutions may be coarsened in time (e.g., every nth time step deleted) without loss of information since the deleted time steps may be recalculated, but as table 4 shows, recalculating these time steps can take considerable time.

Extracts are potentially much smaller than solutions providing substantial data compression if solution time steps can be deleted after extraction. Most extracts are 2-D, 1-D or 0-D. Unless there are a very large number of 1-D or 0-D extracts, the 2-D extracts will dominate storage requirements. If n is the number of nodes in the solutions grid, 2-D extracts require $O(n^{2/3})$ storage. As n becomes large, $O(n^{2/3})$ becomes relatively quite small. Furthermore, a user can control the space consumed by extracts by choosing the number and type of extracts to calculate. These effects can combine to achieve order of magnitude lossy data compression when extracts are substituted for solutions, or as little as a factor of three or four compression. Appendix D contains an equation for calculating extract sizes as a function of solution grid size and the desired extracts along with supporting data. Note that 2-D extracts generated by the marching cubes algorithm tend to be large since the resulting unstructured grids require lists of triangles as well as arrays of nodes.

Table 7 suggests that extract calculations can be a significant fraction of the CPU time a solver requires. This suggests that if extracts for all time must be calculated, extracts should be calculated in batch mode. Alternatively, extracts for individual time steps might use lazy evaluation in hopes of avoiding some calculation. This approach will mean very slow animation rates until all the desired extracts have been calculated. Alternately, extracts can be derived from solutions coarsened in space and/or time to speed calculation at the expense of accuracy. Full resolution calculations can be made in batch once the right extracts have been found. As we will see, conversion of extracts to graphic objects and viewing can be accomplished interactively.

Graphic objects can be implemented by converting extracts to graphic specific data structures, by rendering functions that take extracts and rendering parameters as arguments, and/or a continuum between these extremes. The choice of implementation affects time and space trade-offs between extracts and graphic objects.

If graphic objects are implemented by converting extracts to graphic data structures, rendering time may be decreased by a factor of two to eleven[1] because graphic data structures are optimized for rendering. The storage required for a graphic object is about the same as that required by the extract represented. Graphic vertices require the same space as extract grids; vertices needed to specify arrow plots of vector fields are the same size as vector fields; and graphic RGB values are about the same size as scalar fields. Note that if an extract is deleted, then most rendering flexibility is lost. There is one case where extract grids are much larger than graphic object vertex lists: when unstructured grid connectivity information is required to optimize particle integration.

If graphic objects are implemented as rendering functions taking extracts as arguments, there is no storage penalty for graphic objects, but rendering time is greater than when special graphic data structures are built. Since no data is discarded in this approach, no flexibility is lost.

Scenes are simply sets of graphic objects combined with a small amount of viewing information so there are no space vs. time vs. flexibility trade-offs to make. Note that canned navigations requiring a small amount of storage are needed to repetitively animate graphic objects -- an important capability when trying to understand a complex visualization.

Converting scenes to movies eliminates most flexibility but greatly increases rendering speed in

1. See appendix D for details.

most cases (30 frames a second for video tape). Although the storage requirements of scenes (including the graphic objects) may be much greater or less than the equivalent movie, movies may be stored on video tape with minimal loss of functionality so space trade-offs are largely irrelevant. User control of movies is limited to time controls: forward, backward, faster, slower, and freeze frame.

VI. Discussion

The perceptive reader will notice that there is no discussion of fields over the entire solution grid calculated from fields produced by the solver, e.g., the way Plot3d calculates many functions. This idea was left out to simplify the presentation. The same reader will note that there is no discussion of statistical techniques such as histograms. These could be thought of as grid-less extracts. Again, these were left out to simplify exposition.

This paper has focused on very large time dependent solutions. These data are characterized by a set of 3-D solutions indexed by time. The fact that the index is time is not crucial to the model, although comparisons between widely separate index values become more important. In a more complex situation, one can imagine numerical experiments with two indexes, e.g., Reynolds number and time. In this case, the model is even more useful but database management and director tools become more complex.

Visualization is used not only to understand results, but also to debug solvers. The system characteristics required for unsteady flow visualization are similar to debugging since a solution is converging and is thus unsteady. For debugging, discarding solution time steps is much less objectionable than when visualizing results, so extracts are more useful.

The database may also be useful for archiving numerical experimental results. The database can be pruned to a bare minimum of storage supporting (or invalidating) an experiment's hypothesis and stored on suitable media along with journal articles and solver source code. As large simulations can consume months of a scientists career, it makes sense to keep a substantial, well organized, portable archive.

VII. Future Work

A prototype system based on this model is in development. We are directing prototype development towards solving the problems of particular investigators rather than building a general purpose system. A full featured, general purpose implementation of the model will probably require a moderate sized programming team.

For the most part, this paper has assumed that users choose the extracts to calculate. However, imagine smart sensors that examine solution time steps to find features and generate extracts appropriately. For example, setting an isosurface threshold at the center of the steepest gradient of a scalar field, or creating three orthogonal cutting planes at vector field stationary points. This concept, contributed by Val Watson, is a rich area for future research.

VIII. Summary

We have presented a software model intended as a guide to solving performance and flexibility problems encountered when visualizing the largest time dependent volumetric CFD results. The model consists of a database of solution time steps, extracts, graphic objects, scenes, and movies

along with associated tools. Extracting extracts rather than graphic objects from a solution increases flexibility and achieves substantial lossy data compression. A persistent object database unifies access to all data types and keeps performance metrics for all objects to support time vs. space vs. flexibility trade-offs. The database also maintains object history to simplify repetitive operations and support efficient save state and restore. Although most visualization systems implement parts of the model and can be analyzed in the model's terms, there is no complete implementation to prove the concept.

We hope to see the model used to guide developers building visualization systems. We also hope to see visualization users use the model for system evaluation and to understand requirements. Developers and users should recognize that all portions of the model are important and none should be ignored or left out, although some will be more important than others for a particular application.

Appendix A: Glossary

The **artist tool** is used to create graphic objects from extracts.

An **animation** is a picture that changes over time.

The **camera tool** produces movies by recording the sequence of images produced by a scene.

A persistent object **database** unifies access to all objects and maintains object histories and performance metrics.

The **director tool** produces scenes by choosing a set of graphic objects to animate and a viewpoint and direction navigation through time and space.

An **extractor tool** creates extracts from solutions and/or other extracts.

Extracts are a subset of a solution's nodes or a resampling of the solution domain. In each case, an extract may extend over some or all time steps. Extracts reduce solution dimensionality or reduce the size of the solution's dimensions to achieve massive lossy data compression. Extracts are extracted from solutions or other extracts by extractor tools.

Graphic objects are sets of vertices with rendering information. Graphic objects are derived from extracts by artist tools.

Movies are sequences of images, e.g., video tapes. Movies are created from scenes by camera tools.

A **navigation** is a sequence of viewpoints and view direction used to animate a set of graphic objects in a scene. Navigations may be canned or interactive.

A **scene** is a set of graphic objects and a navigation. A navigation is a viewpoint and viewing direction evolving over time. The graphic objects to animate and the navigation path are chosen by director tools.

A **solution** is a set of sampled scalar and vector fields evolving in time in a volume. Fields are sampled at each node of a computational grid at each of many time steps. Grids may evolve with time. Solution time steps are generated from initial conditions or previous time steps by solver.

Viewpoint is the location of the viewer, or camera.

View direction is the direction a camera is looking from.

Appendix B: Visualization Techniques as Extracts

Most common visualization techniques can be easily modified to extract extracts rather than graphic objects. Table 2 summarizes the characteristics of common visualization techniques. Fields are not discussed because any field may be calculated on any extract, although some make more sense than others. Table 7 gives performance data for some of these extracts.

Table 2: Visualization Techniques

Visualization Technique	Grid Type Produced[a]	Description
Grid Planes	2-D curvilinear	a two dimensional slice of a 3-D field created by holding one index constant and allowing the other two to vary over their whole range. A set of grid planes over time is similar to a 3-D data set and may be examined with some of the same techniques.
Grid Volumes	3-D solution grid type	a three dimensional portion of a 3-D field created by limiting the number of solution grid nodes included, e.g., by limiting the extent of each dimension.
Surface Grid	2-D solution grid type	covers the surface of the vehicle or hardware investigated.
Cut Plane	2-D unstructured	an arbitrarily oriented plane through the data. If one uses marching cubes to generate the cutplane [Kerlick89], the plane is represented by an unstructured grid of triangles.
Isosurfaces	2-D unstructured	the surface where a scalar field has a constant value. Isosurfaces do not always need to have field values calculated for each vertex since the spatial information alone is frequently of great interest.
Tangent Curves	1-D	a curves passing through a point (the seed) that is everywhere tangent to a vector field. Functions on tangent curves are frequently used to control graphic representation, e.g., stream polygons [Schroeder91].
Particle Trace	0-D	the path a massless particle takes when released into a vector field. Note that in a steady vector field, a particle trace is identical to a tangent curve.
Streaklines	0-D	created by continuously emitting particles from seed points. Each time step new grid nodes are created at the seed point.

Table 2: Visualization Techniques

Visualization Technique	Grid Type Produced[a]	Description
Tangent Surface [Hultquist90, 92b]	2-D curvilinear or unstructured	a surface passing through a curve (the rake) that is everywhere tangent to a vector field.
Pixel Plane Ray Casting	2-D regular	casts rays from the eye-point through each pixel into the data volume. Information collected along the ray is deposited in the pixel. Thus, a 2-D field on the viewing plane is produced consisting of RGB or HLS values.
X-Rays [Darmofal91]	2-D	casts rays from an emitter to a detector. Rays process information along their path and deposit a scalar field on the detector.
Gridigration [Globus92a]	2-D curvilinear	a special case of the x-ray model for curvilinear grids where the emitter is an exterior grid plane, the detector is the opposite grid plane, and rays follow grid lines.
Vector Field Isodirection Line [Yamasaki92a]	1-D	a curve where a vector field is unidirectional. Usually a number of directions are used simultaneously to study a vector field.
Vector Field Topology [Helman90]	0-D, 1-D, 2-D	To a first approximation, *vector field topology* consists of zeros (stationary or critical points) in a vector field and the stable and unstable manifolds passing through them. Thus, at each time step there is a unconnected set of points (0-D), a set of tangent curves (1-D) and tangent surfaces (2-D). The eigenvalues and eigenvectors of the gradient tensor at each stationary point may be thought of as a field on the stationary point grid. In this case, the field may only be desired on parts of the grid (the stationary points but not the manifolds).

a. The type of grid at a single time step.

Appendix C: Existing Visualization Systems

Visualization systems may be analyzed in terms of the model. Table 3 describes a sampling of

visualization systems in the model's terms. A short description of each system follows.

Table 3: Unsteady Visualization Systems

System	Solutions	Extracts	Graphic Objects[a]	Scenes[b]	Movies
Buning85 (Plot3d)	steady curvilinear multi-zone, iblank	no	grid planes, isosurfaces, tangent curves		no
Bancroft90 (FAST)	steady, curvilinear multi-zone iblank	most graphic objects are implement ed as extracts and render-ing func-tions	isosurfaces, cut planes, grid planes, grid volumes, vector field topology, tan-gent curves	navigation by key frames and tween-ing. A graphic object database chooses graphic objects to animate.	stop frame to video disk
SGI92 (Explorer)	steady, curvilinear single zone, stretched, and regular	isosurface, grid planes	isosurfaces, cut planes, grid planes, grid volumes, gridigration, splatting		no
Yamasaki92b	unsteady[c], curvilinear single zone	multiple grid planes for all time	grid planes, isosurfaces	simple time con-trols	no
Bryson91	unsteady, curvilinear single zone	no	tangent curves, parti-cles, streak-lines	interactive 3-D vir-tual reality naviga-tion; time freeze frame	no
Haimes91	unsteady, unstructured, curvilinear multiple abutting zones	computatio nal space fields. Maintains database of extracts.	cut planes, tangent curves, parti-cle paths, streaklines, surface grids, x-rays	time controls at source code level	no

a. Only the extracts that can be converted to graphic objects are listed. Otherwise the lists would be too long.
b. All the systems have interactive 3-D navigation of graphic objects, so only additional scene facilities are mentioned.

c. Steady and unsteady refer to the fields. All of these systems only function on static grids.

Yamasaki92b focuses on distributing unsteady visualization between supercomputers and multiple users on separate workstations. Solution data is organized such that k slices are contiguous through all time, thus optimizing viewing by k slice. Graphic objects are passed to a workstation for rendering and navigation. Gerald-Yamasaki notes that transferring graphic objects minimizes network traffic over other possible problem partitions while taking advantage of specialized workstation graphics hardware.

Bryson91 discusses the *Virtual Wind Tunnel* using virtual reality technology to visualize time-dependent flow. Graphic objects are calculated from solutions stored entirely in RAM, which improves performance but limits data set size. The Virtual Wind Tunnel has been distributed between a workstation and a mini-super computer to widen the storage bottleneck [Bryson92].

Haimes91 has developed Visual3 to visualize unsteady 3-D flows. Visual3 has the somewhat unique feature of 3-, 2-, and 1-D windows displaying the same data from different dimensional perspectives. For example, Visual3 might display a cutting plane in 3-D, a 2-D window of the cutting plane color mapped, and a 1-D xy plot of a curve on the cutting plane. Visual3 is a modify-the-source-and-rebuild system.

Appendix D: Storage and CPU Time Data

Solution Data

Table 4: Time and Space for a Few Unsteady CFD Simulations at NAS[a]

Data Set	Nodes	Time Steps[b]	Time Step Size[c] (megabytes)	Total Size[d] (gigabytes)	Disk Time per Time Step[e] (sec)	CPU Time per Time Step[f] (sec)
Smith91	2,800,000	100	56	5.6	1.7 / 15.6	420
GB92	1,400,000	400	28	11.2	0.8 / 7.8	675
Atwood92	1,600,000	500	32	16.0	1.0 / 8.9	480
Chawla92	900,000	9,000	18	162.0	0.5 / 5.0	148

a. For the multi-zone curvilinear grids used by these solvers, each node requires a location (three floating point numbers) and a mask (one integer) for the grid plus five floats for each solution time step. This table assumes four bytes per float or integer, which is appropriate for visualization.

b. The number of solution time steps required for visualization purposes. It is important distinguish between visualization time steps and simulation time steps. Typically there are 5-20 simulation time steps per visualization time step. Solvers must take small time steps for a number of reasons, but the solution changes so little that adjacent time steps are virtually identical to the eye. Note that extracts that do not accumulate error, such as isosurfaces, can use larger time steps than those that do, such as integration. When integrating a curve, error accumulates with each integration step.

c. The grid is static and, therefor, is not included.

d. NAS workstations may have 1-2 gigabytes of disk. The NAS Convex has 100 gigabytes of disk storage.

e. Time to read a time step's data from disk. The two values are the fastest transfers achieved at NAS on the Convex (33 Mbytes/sec. [Miya92]) and SGI workstation (3.6 Mbytes/sec. [Lam92]) respectively.

f. CPU time to calculate a visualization time step on the NAS Cray YMP. Wall clock time is usually much longer due to resource competition with other jobs.

Extract Sizes

Three grid planes, three large marching cubes surfaces (isosurfaces or cutting planes), and a particle cloud containing 20,000 particles are extracted from various size solutions on curvilinear grids. Assume two scalar and one vector fields are in each extract.

•

Table 5: Relationship Between Solution and Extract Time Step Storage Requirements

Solution Grid (nodes)	Solution (megabytes)	Combined Extracts (megabytes)	Extract as a Percent of Solution
500,000	10	5.2	52%
1,000,000	20	8.0	40%
2,000,000	40	12.5	31%
3,000,000	60	16.3	27%
4,000,000	80	19.7	25%
5,000,000	100	22.8	23%
10,000,000	200	36.0	18%

Furthermore, a user can control the space consumed by extracts by the number and type of extracts to calculate. Consider a solution on a 3,000,000 node solution on a curvilinear grid. Table 6 shows storage requirements for various extracts. Note that the marching cubes surfaces consume the most storage:

Table 6: Space Consumed by Different Extracts

Measure	Grid Plane	Small[a] Marching Cubes Surface	Large Marching Cube Surface	10,000 Particles
Size (megabytes)	0.3 / 0.5[b]	0.8 / 1.0	3.3 / 4.0	0.2 / 0.3
Percent of Solution Size	0.5% / 0.8%	1.4% / 1.7%	5.6% / 6.7%	0.3% / 0.4%

a. Marching cubes surfaces can vary greatly in size depending on the number of cells intersected. Appendix A contains some data on this issue.

b. The first number assumes one scalar field on the extract's grid. The second number assumes on vector field on the extract's grid.

Extractor CPU Time

Table 7 provides performance metrics for the extracts and related graphic objects in table 2. Many simplifying assumptions were made generating this table, some of which are described in the

footnotes.

Table 7: Extract Performance[a] Characteristics

Extract	Number in a Scene[b]	Grid Nodes (thousands)	Grid Size (Mbytes)	Scalar/Vector Field Size per Time Step[c] (Mbytes)	Extract Calc/Render CPU Time per Time Step (seconds)	Graphic Object Render CPU Time per Time Step (seconds)
Grid Plane	5	111	1.78	0.44/1.33	NA/7.94	2.43
Cut Plane	5	265	9.38	1.06/3.18	66/26.46	6.61
Isosurface	3	214	7.55	0.86/2.57	53/20.24	5.69
Tangent Curve	100	222	2.66	0.89/2.66	131/0.72	0.55
Vector Field Topology	0D: 88 1D: 265	0D: 0.088 1D: 148	0D: 0 1D: 1.78	0D:0/0 1D:0.59/1.78	0D: 406/0.53 1D: 1041/1.15	0D: 0.49 1D: 0.75
Total	NA	960	23	3.84/11.53	1697/57	16.51
Total w/o topology	NA	811	21	3.25/9.74	250/55	15
Solution[d]	1	1,000	16	4/12/20[e]	150[f]	NA

a. These data were taken using FAST on a 226,800 node data set.[Rizk85] on an SGI 320 VGX with 64 Mbytes of memory. All numbers were linearly extrapolated to simulate a 1,000,000 point data set. Elapsed times were measured with a stop watch.Rendering times were timed for 100 iterations and averaged.
b. This column is the number of each type of extract that might be desired for a visualization. This number, although somewhat arbitrary, is necessary to calculate the values in the performance columns.
c. Assuming one scalar and one vector field for each extract.
d. Added for reference.
e. 20 Mbytes for a typical 5 value (density, momentum vector, energy) CFD solution
f. Solver seconds per visualization time step assuming 10 solver time steps per visualization time step. Speeds are consistent with table 1, i.e., on a Cray YMP.

Graphic Object Issues

Table 8: Extract vs. Graphic Object Sizes for Field Representation

Extract Field	Size per Node (bytes)	Graphic Object	Rendering Data Storage per vertex (bytes)
Scalar	4	RGB Colormap	3 - 4
Scalar	4	Index Colormap	1 - 4
Scalar	4	Contours	Not Applicable
Scalar	4	Height Field	4 - 12
Vector	12	Lines	12
Vector	12	Lines w/Arrows	12+

Incorporate my and FAST rendering time data.

Acknowledgments

E. Raible suggested the term 'extract'. M. Gerald-Yamasaki, R. Haimes, T. Lasinski, D. Lane, K. Miceli, E. Miya, and A. Vaziri reviewed drafts of this paper and provided many helpful comments, some of which lead to wholesale revisions.

References

[Atwood92] C. A. Atwood and W. R. Van Dalsem, "Flowfield Simulation about the SOFIA Airborne Observatory," *30th AIAA Aerospace Sciences Meeting and Exhibit*, January 1992, Reno, Nevada, AIAA 92-0656.

[Bailey86] F. R. Bailey, "Status and Projections of the NAS Program," *Symposium on Future Directions of Computational Mechanics*, ASME, Anaheim, CA, 7-12 December, 1986.

[Bancroft90] G. Bancroft, F. Merritt, T. Plessel, P. Kelaita, R. McCabe, and A. Globus, "FAST: A Multi-Processing Environment for Visualization of CFD," *Proceedings Visualization '90*, IEEE Computer Society, San Francisco (1990).

[Bryson91] S. Bryson and C. Levit, "The Virtual Windtunnel: An Environment for the Exploration of Three-Dimensional Unsteady Flows," *Proceedings IEEE/ACM SIGGRAPH Visualization '91*, 22-25 October, 1991, San Diego, California.

[Bryson92] S. Bryson and M. Gerald-Yamasaki, "The Distributed Virtual Windtunnel", *Proceedings of the IEEE/ACM Supercomputing '92*, November, 1992.

[Buning85] P. G. Buning and J. L. Steger, "Graphics and Flow Visualization in Computational Fluid Dynamics," AIAA-85-1507-CP, AIAA 7th Computational Fluid Dynamics Conference, 15-17 July, 1985, Cincinnati, OH.

[Butler91] D. M. Butler, C. Hansen, organizers, "Visualization '91 Workshop Report: Scientific

Visualization Environments," IEEE/ACM SIGGRAPH Visualization '91, 22-25 October, 1991, San Diego, California.

[Chawla92] K. Chawla and W. R. Van Dalsem, "Numerical Simulation of STOL Operations Using Thrust Reversers," AIAA 92-4254, *AIAA Aircraft Design Systems Meeting*, 24-26 August, 1992, Hilton Head, SC.

[Choi87] D. Choi and C. Levit, "Implementation of a Distributed Graphics System," *International Journal Supercomputing Applications*, Winter 1987, pp 82-95.

[Darmofal91] D. Darmofal, R. Haimes, "Visual Feature Identification for 3-D Data Sets," AIAA-91-1583-CP, *Proceedings AIAA 10th Computational Fluid Dynamics Conference*, Honolulu, Hawaii, June 24-27, 1991.

[Dickenson91] R. R. Dickenson, "Interactive VIsualization of Transient Fields," *Proceedings SPIE/IS&T Symposium on Electronic Imaging Science and Technology*, 24 February - 1 March 1991, San Jose, California, USA.

[Globus91a] A. Globus, "Optree Optimization." SPIE paper 1459-01, *Proceedings SPIE/IS&T Symposium on Electronic Imaging Science and Technology*, 24 February - 1 March 1991, San Jose, California, USA.

[Globus91b] A. Globus, C. Levit, T. Lasinski, "A Tool for Visualizing the Topology of Three-Dimensional Vector Fields," *Proceedings Visualization '91*, IEEE Computer Society, San Diego, CA, 1991.

[Globus92a] A. Globus, "Gridigrator: A Very Fast Volume Renderer for 3D Scalar Fields Defined on Curvilinear Grids," NASA Ames Research Center, NAS Systems Division, Applied Research Branch technical report RNR-92-001, January 1992.

[Globus92b] A. Globus and E. Raible, "13 Ways to Say Nothing with Scientific Visualization," NASA Ames Research Center, NAS Systems Division, Applied Research Branch technical report RNR-92-006, March 1992.

[Globus92c] A. Globus, "Perspectives on the IRIS Explorer Visualization Environment," NASA Ames Research Center, NAS Systems Division, Applied Research Branch technical report RNR-92-021, May 1992.

[GB92] K. Gundy-Burlet, personal communication.

[Goldberg81] A. Goldberg, D. Robinson, and D. H. Ingalls, *Smalltalk-80: The Language and Its Implementation.*

[Haber91] R. B. Haber, B. Lucas, and N. Collins, "A Data Model for Scientific Visualization with Provisions for Regular and Irregular Grids," *Proceedings IEEE/ACM SIGGRAPH Visualization '91*, 22-25 October, 1991, San Diego, California.

[Haimes91] R. Haimes and M. Giles, "VISUAL3: Interactive Unsteady Unstructured 3D Visualization," *29th AIAA Aerospace Sciences Meeting and Exhibit*, January 1991, Reno, Nevada, AIAA 91-0794.

[Haimes92] R. Haimes, "Techniques for Interactive and Interrogative Scientific Volumetric Visualization," submitted for publication.

[Helman90] J. L. Helman and L. Hesselink, "Surface Representation of Two- and Three-Dimensional Fluid Flow Topology," *Proceedings Visualization '90*, IEEE Computer Society, San Fran-

cisco 1990.

[Henderson89] R. L. Henderson and A. Poston, "MSS-II and RASH: a Mainframe UNIX Based Mass Storage System with a Rapid Access Storage Hierarchy File Management System," *Proceedings of the Winter 1989 USENIX Conference*, pp 65-84, USENIX Association, January 1989, San Diego, CA.

[Hibbard88] E. A. Hibbard and G. Makatura, *ARCGraph System User's Manual Version 7.0*, January 1988.

[Hultquist90] J. P. M. Hultquist, "Interactive Numerical Flow Visualization Using Stream Surfaces," *Computing Systems in Engineering 1* (2-4) pp. 349-353.

[Hultquist92a] J. P. Hultquist and E. L. Raible, "SuperGlue: A Programming Environment for Scientific Visualization," *Proceedings IEEE/ACM SIGGRAPH Visualization '92*, October, 1992, Boston, MA.

[Hultquist92b] J. P. Hultquist, "Constructing Stream Surfaces in Steady 3-D Vector Fields," *Proceedings IEEE/ACM SIGGRAPH Visualization '92*, October, 1992, Boston, MA.

[Kerlick89] G. D. Kerlick, "ISOLEV: A Level Surface Cutting Plane Program for CFD Data," NASA Ames Research Center, NAS Systems Division, Applied Research Branch technical report RNR-89-006, March 1992.

[Lam92] T. L. Lam, "Improving File System Performance by Striping," NASA Ames Research Center, NAS Systems Division, Systems Development Branch technical report RND-92-014, April 1992.

[Lang91] U. Lang, R. Lang, and R. Ruehle, "Integration of Visualization and Scientific Calculation in a Software System," *Proceedings IEEE/ACM SIGGRAPH Visualization '91*, 22-25 October, 1991, San Diego, California.

[Legensky90] S. M. Legensky, "Interactive Investigation of Fluid Mechanics Data Sets," *Proceedings Visualization '90*, IEEE Computer Society, San Francisco, 1990.

[Levit91] C. Levit and Denis Jespersen, "Numerical Simulation of Flow Past a Tapered Cylinder," *29th AIAA Aerospace Sciences Meeting and Exhibit*, January 1991, Reno, Nevada, AIAA 91-0751.

[Levit92] Personal communication.

[Lorenson87] W. E. Lorenson and H. E. Cline, "Marching Cubes: a High Resolution 3d Surface Construction Algorithm," *Computer Graphics*, Vol. 21 No. 4, July 1987, pp. 163-169.

[Miya92] E. Miya, personal communication.

[Moran92] P. J. Moran and C. S. Potter, "Tiller: A Tool for Analyzing 4-D Data," *Proceedings SPIE/IS&T Symposium on Electronic Imaging Science and Technology*, 9 -14 February 1992, San Jose, California, USA.

[Plessel91] T. Plessel, *GAS: Graphics Animation System, Version 1.22, User's Manual*, January 1991.

[Rizk85] Y.M. Rizk, S. Ben-Shmuel, "Computation of the Viscous Flow Around the Shuttle Orbiter at Low Supersonic Speeds," *AIAA 23rd Aerospace Sciences Meeting*, Jan. 14-17, (1985) Reno, Nevada, AIAA-85-0168.

[Rogers87] S. E. Rogers, P. G. Buning, F. J. Merritt, "Distributed Interactive Graphics Applica-

tions in Computational Fluid Dynamics," *The International Journal of Supercomputer Applications*, Vol. 1, No. 4, Winter 1987, pp. 96-105.

[Schroeder91] W. J. Schroeder, C. R. Volpe, and W. E. Lorensen, "*The Stream Polygon*: A Technique for 3D Vector Field Visualization," *Proceedings IEEE/ACM SIGGRAPH Visualization '91*, 22-25 October, 1991, San Diego, California.

[SGI92] *IRIS Explorer User's Guide*, Silicon Graphics Computer Systems, Document 007-1369-010, 1992.

[Silver91] D. Silver, M. Gao and N. Zabusky, "Visualizing Causal Effects in 4D Space-Time Vector Fields," *Proceedings IEEE/ACM SIGGRAPH Visualization '91*, 22-25 October, 1991, San Diego, California.

[Smith89] M. H. Smith, W. R. Van Dalsem, R. C. Dougherty, and P. G. Buning, "Analysis and Visualization of Complex Unsteady Three-Dimensional Flows," *27th AIAA Aerospace Sciences Meeting and Exhibit*, January 1989, Reno, Nevada, AIAA 91-0794.

[Smith91] M. Smith, K. Chawla, and W. Van Dalsem, "Numerical Simulation of a Complete Stovl Aircraft in Ground Effect," AIAA-91-3293, *AIAA 9th Applied Aerodynamics Conference*, 23-25 September 1991, Baltimore, MD.

[Treinish90] L. A. Treinish, "SIGGRAPH '90 Workshop Report: Data Structures and Access Software for Scientific Visualization."

[Upson89] C. Upson, T. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, A. van Dam, "The Application Visualization System: A Computational Environment for Scientific Visualization," *IEEE Computer Graphics and Applications*, July 1989, pp. 30-41.

[Walther91] S. S. Walther and R. L. Peskin, "Object-Oriented Data Management for Interactive Visual Analysis of Three Dimensional Models," *Proceedings SPIE/IS&T Symposium on Electronic Imaging Science and Technology*, 24 February - 1 March 1991, San Jose, California, USA.

[Wavefront91] *The Data Visualizer User's Guide*. Wavefront Technologies, Inc.

[Yamasaki92a] M. J. Gerald-Yamasaki, "Visualizing the Isodirection Lines of a Vector Field," NASA Ames Research Center, NAS Systems Division, Applied Research Branch technical report RNR-92-017, March 1992.

[Yamasaki92b] M. J. Gerald-Yamasaki, "Interactive and Cooperative Visualization of Unsteady Fluid Flow," NASA Ames Research Center, NAS Systems Division, Applied Research Branch technical report RNR-92-018, March 1992.